Using Time-Series Models for Heart Rate Prediction

Name: Yamini Sai Ratnam Peketi

Student ID: 21147160

Course: MSC AI

Module: Applied Ai

Contents

Using Time-Series Models for Heart Rate Prediction

Introduction:

Heart rate prediction is a crucial task in healthcare that can provide valuable information about a patient's overall health. In this blog post, we discuss the heart rate prediction task and analyze three models, namely ARIMA, SARIMAX, and Exponential Smoothing.

Data-Set Description:

This dataset contains((PT_Train.csv) heart rate and respiration rate data from a Lifetouch device, as well as SpO2 and pulse data from an oximeter. The data was collected at 1-minute intervals over a period of approximately 4 hours. The dataset has 226 entries and 5 columns.

```
df.head(5)
```

	Timestamp (GMT)	Lifetouch Heart Rate	Lifetouch Respiration Rate	Oximeter SpO2	Oximeter Pulse
0	17/08/2015 15:09	139	41	83.450262	126.335079
1	17/08/2015 15:10	144	40	92.000000	140.000000
2	17/08/2015 15:11	140	42	89.000000	144.000000
3	17/08/2015 15:12	138	45	93.000000	141.000000
4	17/08/2015 15:13	133	42	94.000000	134.000000

Fig1: Data Frame

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 226 entries, 0 to 225
Data columns (total 5 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Timestamp (GMT)             226 non-null    object
 1   Lifetouch Heart Rate        226 non-null    int64
 2   Lifetouch Respiration Rate  226 non-null    int64
 3   Oximeter SpO2               191 non-null    float64
 4   Oximeter Pulse              191 non-null    float64
dtypes: float64(2), int64(2), object(1)
memory usage: 9.0+ KB
```

Fig2: Data Frame

Data Pre-Processing:

In the heart rate prediction task, we cleaned the dataset, handled missing values, normalized the data, and addressed the outliers.

+ Code + Text

Add text cell

```
#Impute missing values with a suitable value (e.g. the mean)
df["Oximeter SpO2"].fillna(df["Oximeter SpO2"].mean(), inplace=True)
df["Oximeter Pulse"].fillna(df["Oximeter Pulse"].mean(), inplace=True)

#here we are replacing the negative values
df=df.replace(-1, np.nan, inplace=False)

#handling-outliers: here for the heart rate we have consider the values which are between 40-200 as we know that
df = df[(df['Lifetouch Heart Rate'] > 40) & (df['Lifetouch Heart Rate'] < 200)]
```

Fig3: Data Pre-Processing

We then split the dataset into training and testing sets to evaluate the model's performance.

Comparison of the Models:

In our heart rate prediction task, we checked for stationarity in the dataset before applying any models.

```
# check stationarity of the series
def check_stationarity(series):
  statistic, p_value, n_lags, critical_values = sm.tsa.stattools.kpss(series)
  print(f'p value: {p_value}')
  print(f'Result: The series is {"not " if p_value < 0.05 else ""}stationary \n')
```

Fig4: Checking the stationary of the series

To compare the performance of the three models, we used several metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), R-squared, AIC, and BIC. Our analysis showed that the Exponential Smoothing model with triple exponential smoothing provides the best fit and performed the best in all metrics we evaluated.

Model	MSE	MAE	R-squared	AIC	BIC
ARIMA	111.74	3.69	0.13	1302.33	1309.08
ARIMA	22.85	3.12	0.82	1309.31	1322.85
SARIMAX	111.82	3.70	0.13	1291.30	1298.04
Exponential Smoothing	23.10	3.06	0.82	688.46	695.23
Exponential Smoothing	23.10	3.06	0.82	692.46	706.00
Exponential Smoothing	21.69	3.06	0.83	718.74	799.97

Fig 5: Metrics Evaluation

Analysing the Results and choosing the best Model:

Choosing the best model is crucial for machine learning tasks, and our analysis shows that the Exponential Smoothing model is the best fit for heart rate prediction. This model uses a combination of the trend, seasonal, and error components of the dataset to predict the heart rate accurately.

```
predictions_tripple_fit

218    140.044319
219    140.091807
220    139.633739
221    140.973178
222    140.150160
223    139.529062
224    140.867395
225    139.797353
226    138.258035
227    138.654237
228    138.880642
229    137.248056
230    138.099101
231    138.475176
232    139.639585
233    136.310522
234    138.108488
235    137.550339
236    138.001289
237    140.094550
dtype: float64

print(single_fit.summary())
```

Fig6: Predictions

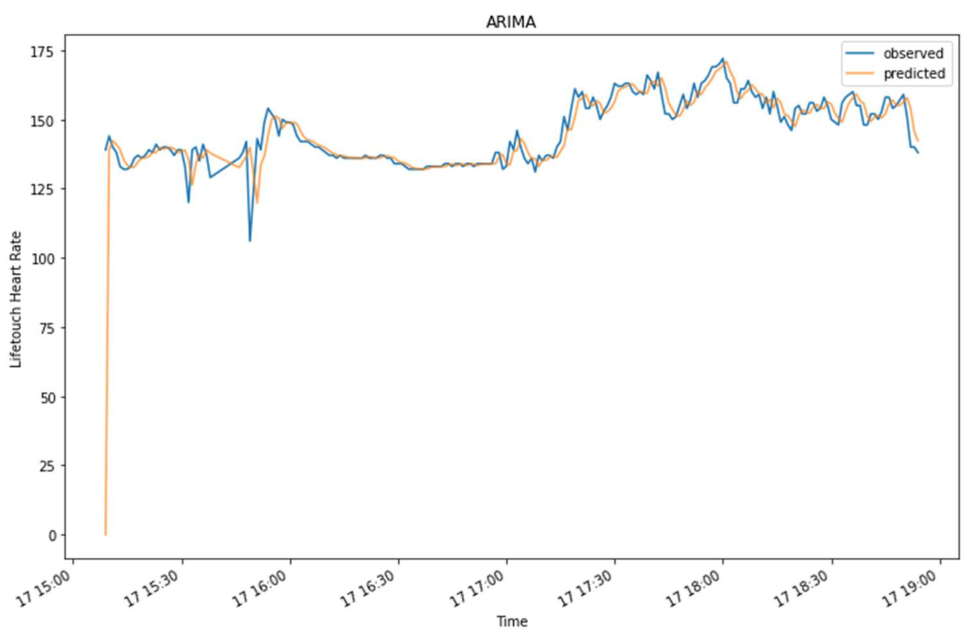

Fig7: Arima Model Plot Observed v/s Predicted

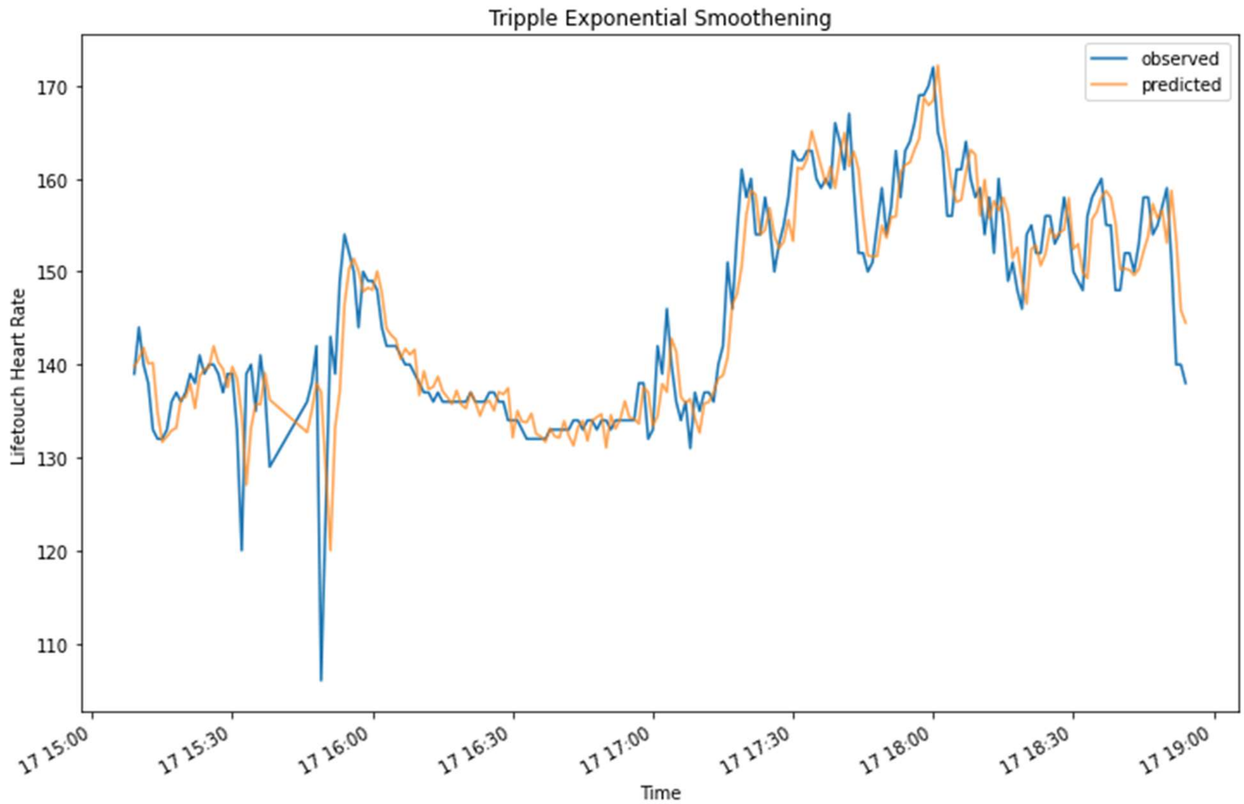

Fig 8: Smoothening Model plot

Conclusion:

In conclusion, our analysis of the three models, ARIMA, SARIMAX, and Exponential Smoothing, showed that the Exponential Smoothing model with triple exponential smoothing provides the best fit and accurate predictions. This model can be used in real-time applications to monitor a patient's health and provide valuable insights to healthcare professionals.

Word Count: 304 (Excluding contents,images,links)

Code:
https://colab.research.google.com/drive/1D8iCFyzansNJ9sJsETt95gLiXjczb2ej?usp=sharing

Dataset Link: https://raw.githubusercontent.com/yamini542/Applied-AI/main/PT_Train.csv